

Received

DEC 08 1999

Director's Office

Group 2700

ORG #: Group 2700

DATE: Oct. 2, 1984

ITEM #: 84051

AUTHOR: Roger Knights

SUBJECT: Date-Handling Functions

COBOL COMMITTEE
WORKING PAPER

A family of date-handling functions was authorized at the 83-11 meeting (p. 35 of minutes). I hope this paper will get the ball rolling, even though its direction may well need considerable correction. I suggest handling three main date formats.

1. **YR-DAY** and **YEAR-DAY** are inferred from lengths of 5 and 7 (using PIC 9 (USAGE DISPLAY) or X). [2- & 4-digit years respectively]. (The leftmost 2 digits of year, if omitted in an argument, are assumed to equal those of the century in **CURRENT-DATE**.)
2. **YR-MO-DA** and **YEAR-MO-DA** are inferred similarly from lengths of 6 and 8. "Year first" is the standard internal Cobol date sequence, and is also the ISO standard all-numeric date format. (Hyphens are the official ISO separators.)

If a 6- or 8-digit date argument is in "customary" American or European format, a parenthetical suffix to the argument may be used, e.g. "(MDY)" or "(DMY)". (Internally these would simply resequence the fields of the argument before invoking the corresponding year-first function.) For symmetry the default suffix "(YMD)" may also be used.

If it is desired to have the value returned be in "customary" format, the four functions **DA-MO-YR/YEAR** and **MO-DA-YR/YEAR** may be used. (Internally they would simply invoke **YR/YEAR-MO-DA** and resequence its output, just as the "YR" functions simply invoke the "YEAR" functions and chop off their left two digits. These all spare the coder the trouble of having to do it himself.)

3. **ABS-DATE** is inferred from PIC 9(9). (Or perhaps it could also be inferred from the combination of 9(7) and a USAGE of **BINARY** or **PACKED** (to avoid confusion with format 1), since it would rarely be used externally, and since this would allow storage in 3 bytes.)

The absolute date for Jan. 1, 1990 is 2,447,893. Absolute date 0,000,001 was Jan. 1, 4713 B.C. I attach a copy of a letter from Computerworld by Richard L. Conner containing more information on the absolute format. (Conner has given me some tips on this paper.) The absolute format would be a useful base for users of Islamic & other non-Gregorian calendars to work with, as well as being the

1000100

DEC 08 1999

most efficient method for Westerners who take ~~the Director's Office~~ to convert to it, so it should be allowed even though ~~not~~ not as common as the other two.

Note that the modulus 7 of any absolute date, plus 1, yields the day-of-the-week number returned by CURRENT-DATE. I've used the term "absolute", which is its name in astronomy, rather than "Julian", to avoid confusion in those shops that use the latter term as a synonym for YR-DAY.

The date-format of the arguments is inferred from their length or data-class, as outlined above. Fortunately, they all differ. As mentioned, the 6- and 8-digit arguments may be qualified by "(DMY)" and "(MDY)".

The first argument of all the 13 functions I suggest may be in any of the three date formats mentioned above. This symmetry should make things easy to remember. (Although it's not necessary to invoke a function to do date math on absolute dates, which can be added and subtracted directly, I suggest allowing such absolute arguments for symmetry.)

The argument(s) and the format returned by the first 9 functions are indicated below. The functions' names indicate the values returned.

Names	Arg-1	Arg-2	Arg-3	Returns
YR-DAY	Any of the three date formats.	"PLUS" or "MINUS".	9999 thru 9 days (optionally signed).	9(5) [9(7) = 'YEAR-DAY']
YR-MO-DA		"TO" X	Any of the 3 date formats.	9(6) [9(8) = 'YEAR-MO-DA']
ABS-DATE				9(9)
DAYS-FROM				S9(4)
IF-DATE	<rel. oper. >			true or false

* [Or MO-DA-YR/YEAR or DA-MO-YR/YEAR.]

Here are some examples. If only one argument is specified (as in the first six functions below), only a format conversion without addition or subtraction is performed. The format of arguments is spelt out beneath or alongside them, in brackets.

Function	Args.	Returns
1. YR-DAY	(840201) [YR-MO-DA]	84032
2. YR-MO-DA	(840201) [YR-DAY]	840201
3. YR-DAY	(00000001) [ABS] [I suggest such BC dates return 0 or all 9s--else dates will have to return a leading space or a minus.]	9999999

Function	Args.	Returns	Received
4.	ABS-DATE (19830101) [YEAR-MO-DA]	002445336	DEC 08 1999
5.	ABS-DATE (83001) [YR-DAY]	002445336	Director's Office Group 2700
6.	ABS-DATE (2445336) [ABS] [Should this sort of thing be forbidden? --Args. of the same format as the result, I mean.]	002445336	
	Here are the 1st 3 functions above, with 3 arguments; DAYS-FROM and IF-DATE always take 3 arguments		
7.	YEAR-DAY (83362 PLUS 5) [YR-DAY]	1984002	
8.	YEAR-DAY (83001 PLUS <u>365.2524 * 5</u>) [YR-DAY]	1988001	[Expressions are OK in a numeric argument--useful. To avoid any ambiguity, PLUS & MINUS in arg-2 are spelt out.]
9.	YR-MO-DA (840101 MINUS 2) [YR-MO-DA]	831230	
10.	YR-DAY (840105 PLUS 395) [YR-MO-DA]	85034	
	[Extra day in leap year affected result.]		
11.	DAYS-FROM (850203 TO 840105) [Both in YR-MO-DA]	-395	
	[Negative value returned since 2nd arg. is lower; Both args. must have same format else 9999 returned.]		
12.	IF-DATE (84001 = 010184 (DMY)) [YR-DAY] [DA-MO-YR]	true	

A fourth argument might be added (eventually, if necessary) to the functions above: VTOC IS <00-98>, where VTOC stands for "Virtual Turn of the Century". This would cause all dates with years below the VTOC year to be considered to be higher than the year ending in 99. A VTOC of 00 could be coded as a placeholder to indicate that it should be modified at a future date, but it would be ignored by the compiler. By setting the VTOC to some year ahead of all dates in the input, the turn of the century could be "gotten around". Arithmetically lower but temporally higher 2-digit years in the new century would be treated as higher by the function. After all dates in the input had moved into the new century (so that arithmetic comparisons would return the desired results again) the VTOC operand could be reset to 00.

A book published a few months ago (COMPUTERS IN CRISIS By J. & M. MURRAY, WHICH I'LL BRING TO THE MEETING - ATTACHED IS REVIEW IN CW), is devoted entirely to the mess the turn of the century is going to cause. It estimates the conversion cost as being in the billions. If this VTOC argument could save a small percentage of that, it would be worth it. This is a much more modest proposal than the method I suggested last year, which would have retrofitted existing code with a ~~new~~ clause.

Received

DEC 08 1999

Director's Office
Group 2700

The last four functions, below, are the simplest. They can take only one argument, and what they return is obvious from their names.

Names	Arg-1	Returns
DAY-S-IN-MONTH	Any of the 3 date formats	99 (1-12)
DAY-NO	OR:	9 (1-7)
DAY-NAME		A(9)
MONTH-NAME		A(9)

The coder can truncate DAY-NAME & MONTH-NAME's returned values to the length of his target item. If he wants unequal-length abbreviations he can do it himself, later. (Or he can propose a fancier function to the CC. Or resubmit John Piggott's Picture symbol L, which truncates trailing spaces within a picture, so that the day of a month would always follow the month name with just one space.)

The coder could achieve the effect of these two NAME functions by defining and searching a table himself, after converting the dates (if necessary) with one of the first three functions. But I think the directness of the NAME functions is useful and stylistically OK. If most common tasks could be identified and functional-ized, the language would be higher level and coding would be largely a matter of slapping functional "boilerplate" together, rather than traditional "programming". This is the "reusable code" idea, which seems to be a coming thing.

Even though there is some overlap in all these functions, it is better to have that and let the functions internally invoke one another (when necessary) than to squeeze such redundancy out and force the coder to nest his functions two or three deep, which defeats the purpose of the higher level approach that functions give. (An example of such invocation is that the two MONTH functions above would have to invoke YR-NO-DA if their argument were a date in YR-DAY or absolute format, and the two DAY functions would have to invoke ABS-DATE if not passed an absolute date argument.)